**Phase-V:**

**SMART WATER FOUNTAIN WITH IoT**

**Thirukumaran M Vasanth S Nitish Kumar S Vignesh B Ramprasath G**

**Mentor: Mrs. R Keerthana**

**1. Introduction**

The "Smart Water Fountain with IoT" project is aimed at creating an intelligent smart water fountain that leverages Internet of Things (IoT) technology to enhance hygiene and user experience. This project utilizes various sensors to monitor fountain conditions and implement automated actions such as flushing and periodic cleaning. By doing so, it aims to provide a cleaner and more efficient public fountain experience.

**2. Objectives**

The " Smart Water Fountain with IoT” project aims to create an intelligent public fountain that enhances hygiene and user experience through the use of IoT technology. The primary objectives of this project are as follows:

* Improve cleanliness and hygiene in public fountains.
* Enhance user experience by automating essential fountain functions.
* Optimize resource usage, such as water, by using smart sensors and automation.
* Provide real-time information to users about fountain conditions.
* Implement a scalable and reliable IoT system for public fountains.

**3. Data Input:**

1. Occupancy Detection: PIR motion sensors monitor motion at key locations in the fountain and send signals to the control module when someone enters.

2. Temperature-Based Entry Detection: A DHT22 temperature sensor detects a temperature increase of 2°C or more, indicating the entry of a person.

3. Humidity Sensor: A DHT22 humidity sensor tracks humidity levels and triggers actions when the humidity surpasses a defined threshold (e.g., 60%).

**3.1. Data Processing**:

1. Occupancy and Entry Detection: The control module processes signals from occupancy and temperature sensors, incrementing the occupancy counter when both sensors confirm entry.

2. Automatic Flushing: When occupancy is detected, the control module initiates automatic flushing, ensuring cleanliness after fountain use.

3. Periodic Cleaning: A counter tracks the number of occupants, and the control module triggers automatic cleaning after a predefined interval (e.g., every 3 persons).

4. Humidity-Based Flushing: The control module continuously monitors humidity levels and triggers flushing when humidity exceeds a predefined threshold.

**3.2 Code and Functionality:**

1. Sensor Data Reading: The Python code reads data from occupancy, temperature, and humidity sensors, providing essential input.

2. Flushing Logic: Automatic flushing is implemented based on occupancy detection, enhancing cleanliness.

3. Automatic Cleaning: Periodic cleaning is initiated to maintain high hygiene standards.

4. Humidity-Based Flushing: Flushing can also be triggered based on humidity levels, enhancing user comfort.

5. MQTT Communication: MQTT is used for remote control and monitoring of the system, facilitating real-time interaction.

6. Sensor Data Display: The Python code interfaces with a digital display, updating users with occupancy status, temperature, and humidity levels.

**3.3 Digital Display:**

- A digital display connected to the control module informs users of the fountain's status, including occupancy (Vacant/Occupied), real-time temperature (e.g., "Temperature: 25°C"), and current humidity level (e.g., "Humidity: 55%"). This information is dynamically updated for user convenience.

In conclusion, this project successfully combines IoT technology, sensor data, and Python code to create a "Smart Public Fountain" that offers enhanced hygiene, user experience, and efficient fountain operations. Users are provided with real-time information while automated systems maintain cleanliness and comfort.

**4.** **The IoT device setup:**

The following devices are used.

* Occupancy Sensor: A Passive Infrared (PIR) motion sensor placed at strategic locations in the fountain to detect occupancy.
* Temperature Sensor: A DHT22 temperature sensor placed centrally to measure ambient temperature and detect entry based on a temperature rise.
* Humidity Sensor: A DHT22 humidity sensor placed centrally to monitor humidity levels.
* Digital Display: A display connected to the control module to provide real-time information to users.
* Raspberry Pi: The control module responsible for processing data and controlling fountain functions.

**5.Platform Development**

The project involves several key components for platform development:

Data Input: Real-time data is obtained from occupancy, temperature, and humidity sensors placed in the fountain. These sensors continuously monitor the environment and provide essential input for the system.

Data Processing: Data from the sensors is processed by the control module in real-time to make informed decisions regarding fountain operations. This includes occupancy and entry detection, automatic flushing, periodic cleaning, and humidity-based flushing.

Code Implementation: Python is used to develop the code that interfaces with sensors, controls actuators, and handles communication via MQTT.

MQTT Communication: The system uses MQTT for communication, enabling remote control and monitoring. This is crucial for real-time interaction with the fountain system.

**6. Algorithm:**

The Python code plays a central role in the project. It includes the following functionalities:

* Sensor Data Reading: The code reads data from the occupancy, temperature, and humidity sensors, providing the foundational input for the system.
* Flushing Logic: Automatic flushing is initiated based on occupancy detection, ensuring a clean environment for the next user.
* Automatic Cleaning: Periodic cleaning is triggered after a predetermined number of occupants to maintain high hygiene standards.
* Humidity-Based Flushing: The system can also trigger flushing based on humidity levels to ensure user comfort.
* MQTT Communication: The code facilitates communication with the MQTT server, enabling remote control and monitoring.
* Sensor Data Display: To keep users informed, the code interfaces with a digital display, presenting real-time information about occupancy, temperature, and humidity levels.

**7. Code Implementation**

import paho.mqtt.client as mqtt

import RPi.GPIO as GPIO

import Adafruit\_CharLCD as LCD

import time

# Configure GPIO pins for actuators and display

GPIO.setmode(GPIO.BCM)

FLUSH\_PIN = 17 # Pin for flushing

CLEANING\_PIN = 18 # Pin for cleaning (e.g., UV lights or cleaning robot)

DIGITAL\_DISPLAY\_PIN = 19 # Pin for the digital display

GPIO.setup(FLUSH\_PIN, GPIO.OUT)

GPIO.setup(CLEANING\_PIN, GPIO.OUT)

GPIO.setup(DIGITAL\_DISPLAY\_PIN, GPIO.OUT)

# Set up MQTT client for IoT communication

MQTT\_BROKER = "your\_broker\_address"

MQTT\_TOPIC\_SENSORS = "fountain/sensors"

MQTT\_CLIENT\_ID = "fountain\_pi"

client = mqtt.Client(MQTT\_CLIENT\_ID)

client.connect(MQTT\_BROKER, 1883)

# Sensor reading functions (replace with actual sensor code)

def read\_temperature\_sensor():

# Replace with code to read temperature sensor

# Example: temperature = read\_actual\_temperature\_sensor()

temperature = 25.5 # Replace with actual sensor reading

return temperature

def read\_humidity\_sensor():

# Replace with code to read humidity sensor

# Example: humidity = read\_actual\_humidity\_sensor()

humidity = 50.5 # Replace with actual sensor reading

return humidity

def read\_occupancy\_sensor():

# Replace with code to read occupancy sensor

# Example: occupancy = read\_actual\_occupancy\_sensor()

occupancy = True # Replace with actual sensor reading (True for occupied, False for vacant)

return occupancy

# Flushing function

def flush\_toilet():

GPIO.output(FLUSH\_PIN, GPIO.HIGH) # Activate the flush mechanism

time.sleep(2) # Adjust as needed

GPIO.output(FLUSH\_PIN, GPIO.LOW) # Deactivate the flush mechanism

# Cleaning function (example for periodic cleaning)

def clean\_fountain():

# Start the cleaning mechanism here

print("Fountain is being cleaned.")

GPIO.output(CLEANING\_PIN, GPIO.HIGH) # Activate cleaning mechanism

time.sleep(5) # Adjust as needed

GPIO.output(CLEANING\_PIN, GPIO.LOW) # Deactivate cleaning mechanism

# Digital display function

def display\_message(message):

# Control the digital display to show the message

print("Display message:", message)

# Initialize the LCD display

lcd = LCD.Adafruit\_CharLCDPlate()

# Function to update the digital display with a message

def update\_display(occupancy, temperature, humidity):

lcd.clear() # Clear the display

# Display occupancy status

if occupancy:

lcd.message("Status: Occupied\n")

else:

lcd.message("Status: Vacant\n")

# Display temperature and humidity

lcd.message(f"Temp: {temperature}C\n")

lcd.message(f"Humidity: {humidity}%")

# Example usage

if \_\_name\_\_ == "\_\_main\_\_":

# Replace these with actual sensor readings

occupancy = True # Example occupancy status

temperature = 25.5 # Example temperature reading

humidity = 55.0 # Example humidity reading

update\_display(occupancy, temperature, humidity)

# Keep the display on for a few seconds (adjust as needed)

time.sleep(5)

# Clear the display

lcd.clear()

# MQTT message handler

def on\_message(client, userdata, msg):

# Process incoming MQTT messages

payload = msg.payload.decode("utf-8")

# MQTT message handler

def on\_message(client, userdata, msg):

# Process incoming MQTT messages

payload = msg.payload.decode("utf-8")

# Define a list of valid commands you want to handle

valid\_commands = ["flush", "clean", "other\_command"]

if payload in valid\_commands:

if payload == "flush":

# Implement logic to trigger a manual flushing

manual\_flushing()

elif payload == "clean":

# Implement logic to manually start a cleaning cycle

manual\_cleaning()

# Add more elif conditions for other commands as needed

def manual\_flushing():

# Implement logic to manually trigger flushing

# For example, you can activate the flush mechanism for a certain duration

GPIO.output(FLUSH\_PIN, GPIO.HIGH)

time.sleep(2) # Adjust the duration as needed

GPIO.output(FLUSH\_PIN, GPIO.LOW)

def manual\_cleaning():

# Implement logic to manually initiate cleaning

# This might involve starting the cleaning mechanism

print("Manual cleaning has been initiated.")

GPIO.output(CLEANING\_PIN, GPIO.HIGH)

time.sleep(5) # Adjust the duration as needed

GPIO.output(CLEANING\_PIN, GPIO.LOW)

# Subscribe to MQTT commands

client.subscribe("fountain/commands")

client.on\_message = on\_message

client.loop\_start()

# Subscribe to MQTT commands

client.subscribe("fountain/commands")

client.on\_message = on\_message

client.loop\_start()

# Counters for occupancy and cleaning

occupancy\_counter = 0

cleaning\_interval = 3 # Clean automatically every 3 persons

try:

while True:

temperature = read\_temperature\_sensor()

humidity = read\_humidity\_sensor()

occupancy = read\_occupancy\_sensor()

# Detect person based on temperature rise

if temperature > 25:

occupancy\_counter += 1

# Check if it's time to trigger automatic cleaning

if occupancy\_counter >= cleaning\_interval:

clean\_fountain()

occupancy\_counter = 0

# Flushing logic based on humidity

if humidity > 60:

flush\_toilet()

# Publish sensor data to MQTT topic

sensor\_data = {

"temperature": temperature,

"humidity": humidity,

"occupancy": occupancy

}

client.publish(MQTT\_TOPIC\_SENSORS, str(sensor\_data))

# Display sensor data on the digital display

display\_message(f"Temp: {temperature}°C, Humidity: {humidity}%, Occupancy: {occupancy}")

time.sleep(10) # Adjust the interval as needed

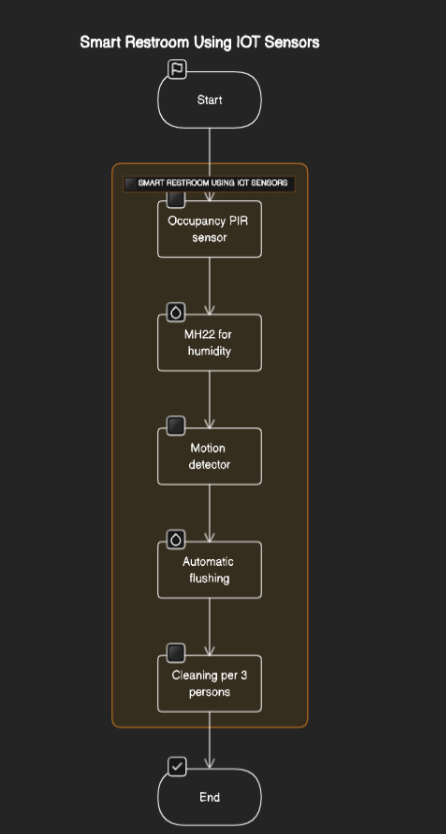
except KeyboardInterrupt:

GPIO.cleanup()

client.disconnect()

client.loop\_stop()

**8. Flow Diagram**

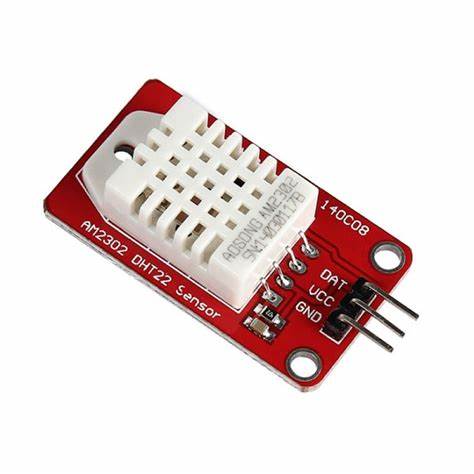


**9. IoT devices**

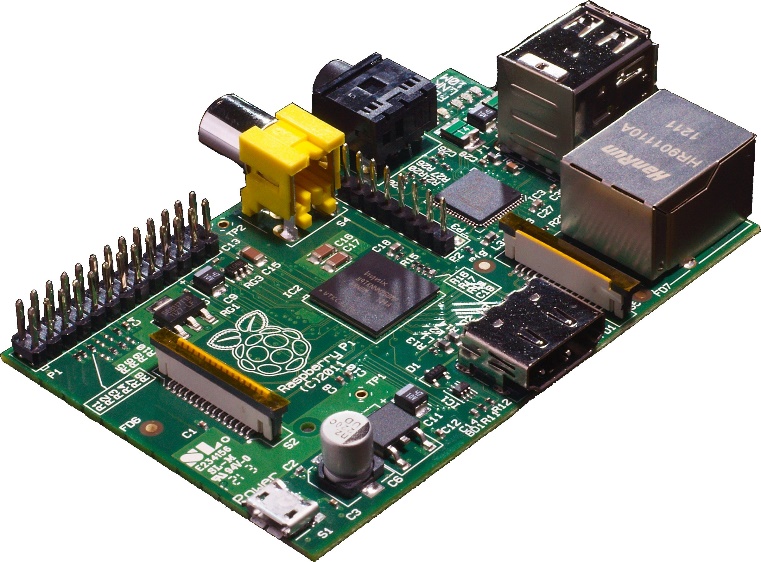
**i) PIR (Passive Infrared sensor)**



**ii)DHT22 sensor (for human and humidity detection )**



**iii)Rasberry pi**

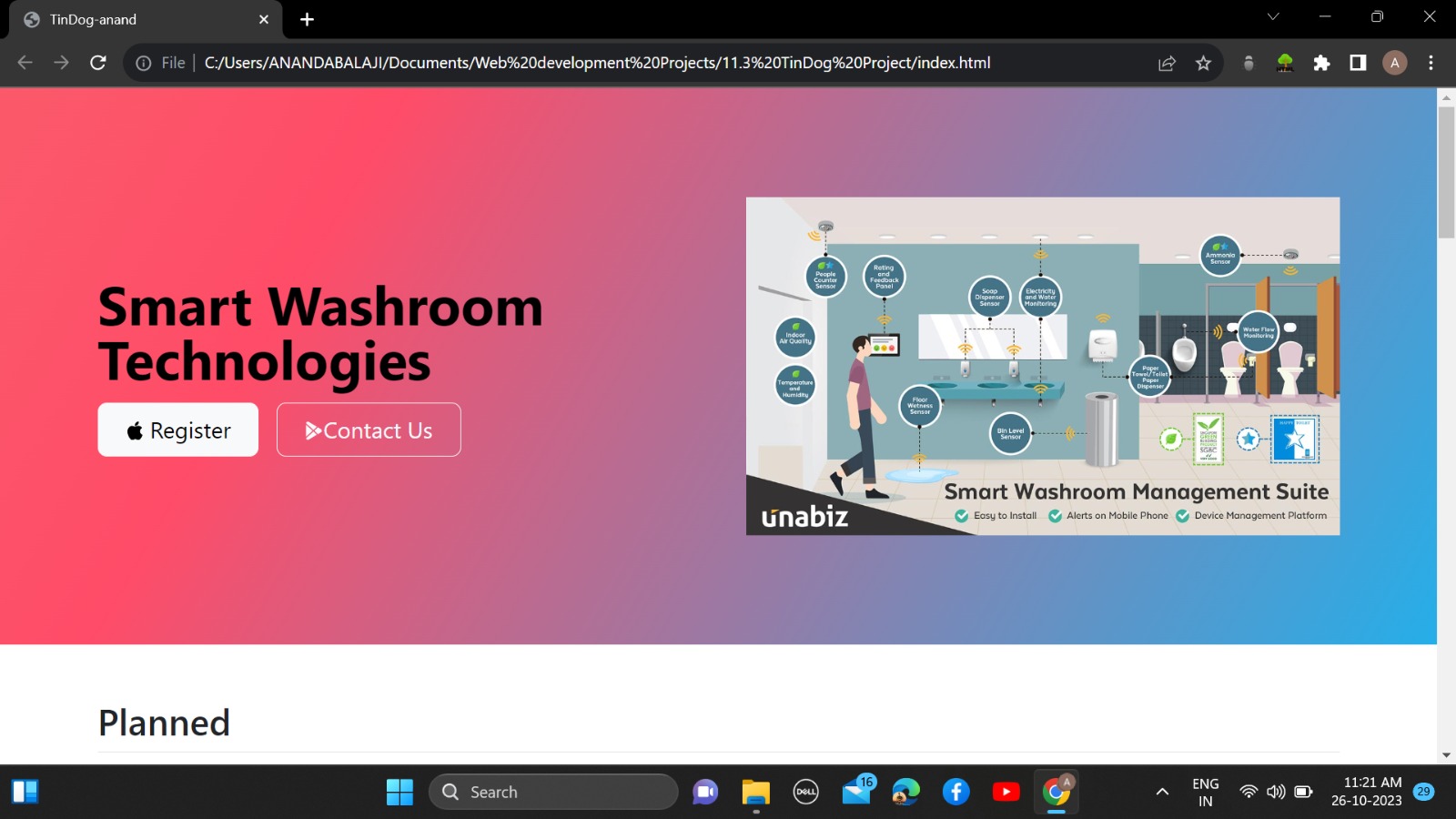


**10. Data Sharing Platform:**

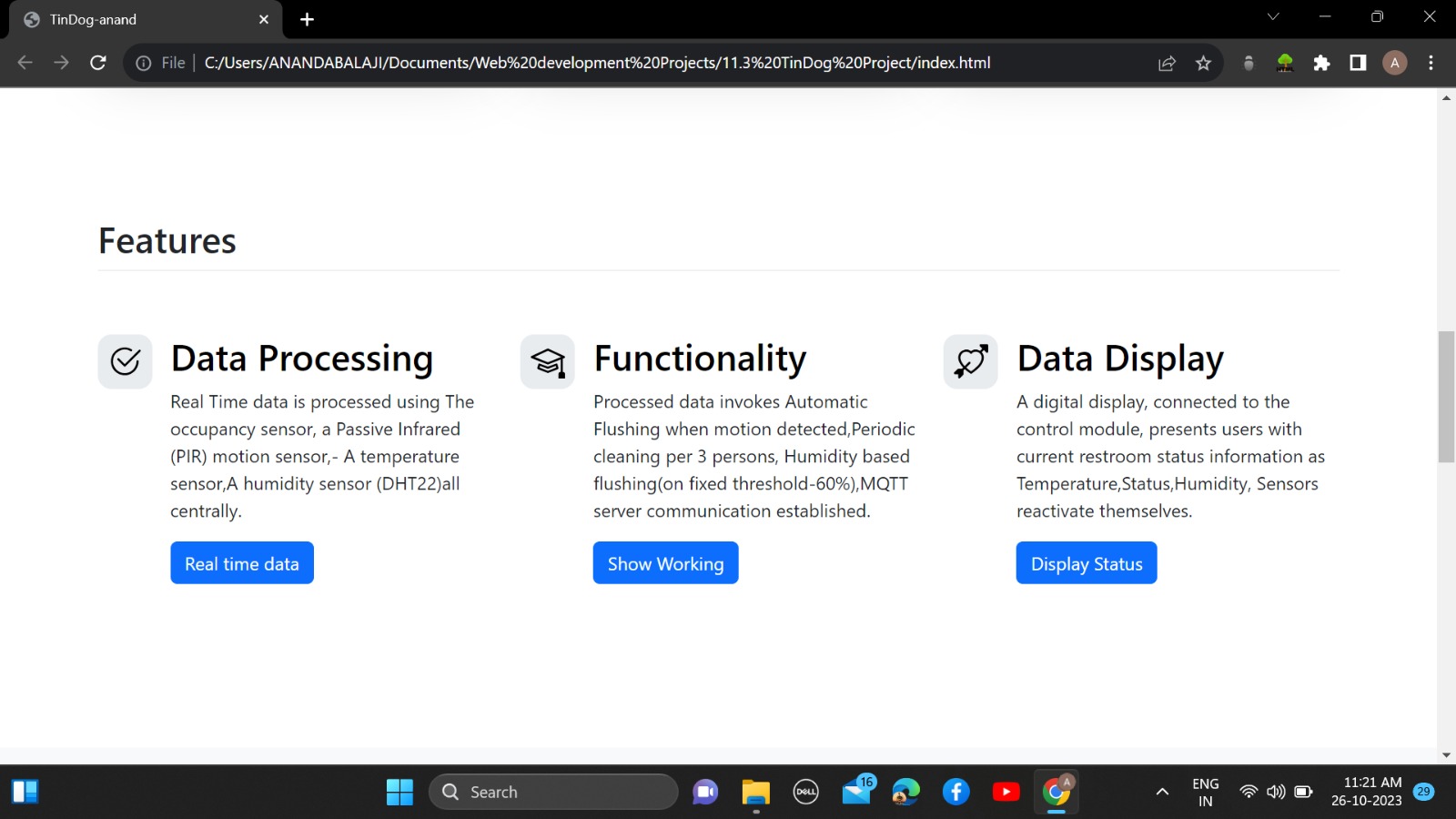
DEVELOPMENT PART:

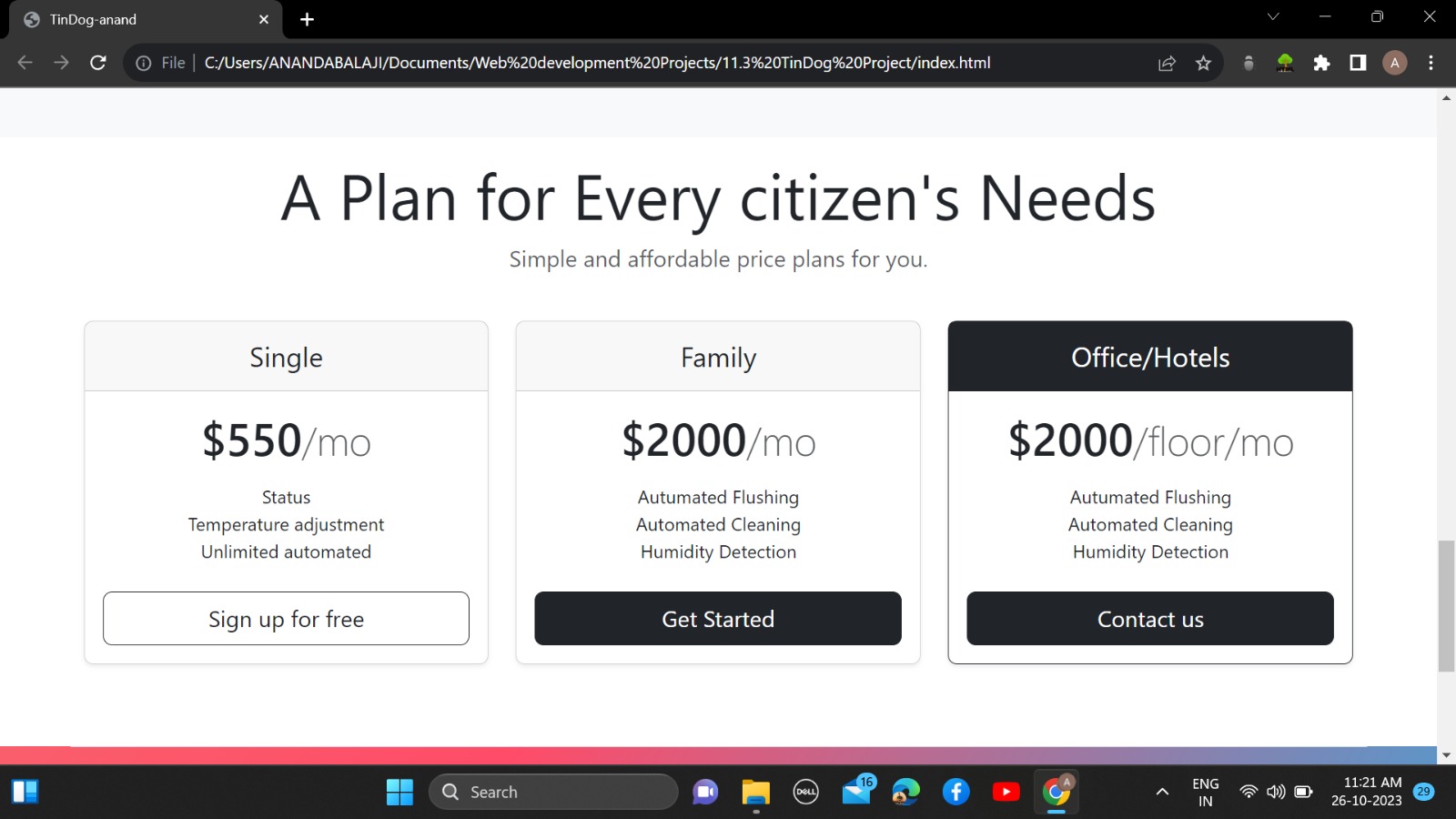
Designed mobile apps for iOS and Android platforms that provide users with access to realtime fountain information. This contains the information regarding the various kinds of services provided by Smart Washroom technologies for arranging fountains using IoT devices.

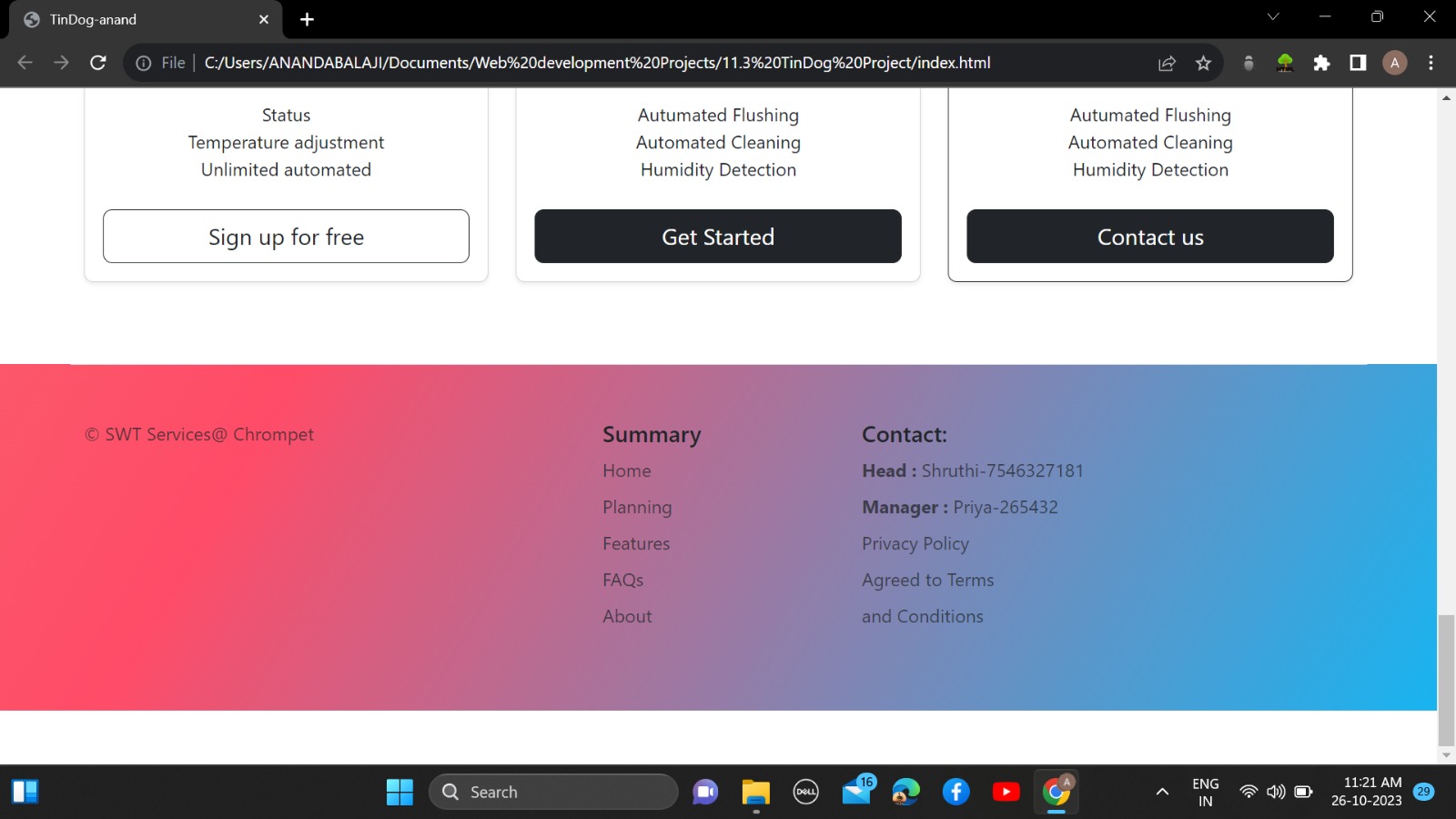
**Screenshots of the mobile App implementation part:**

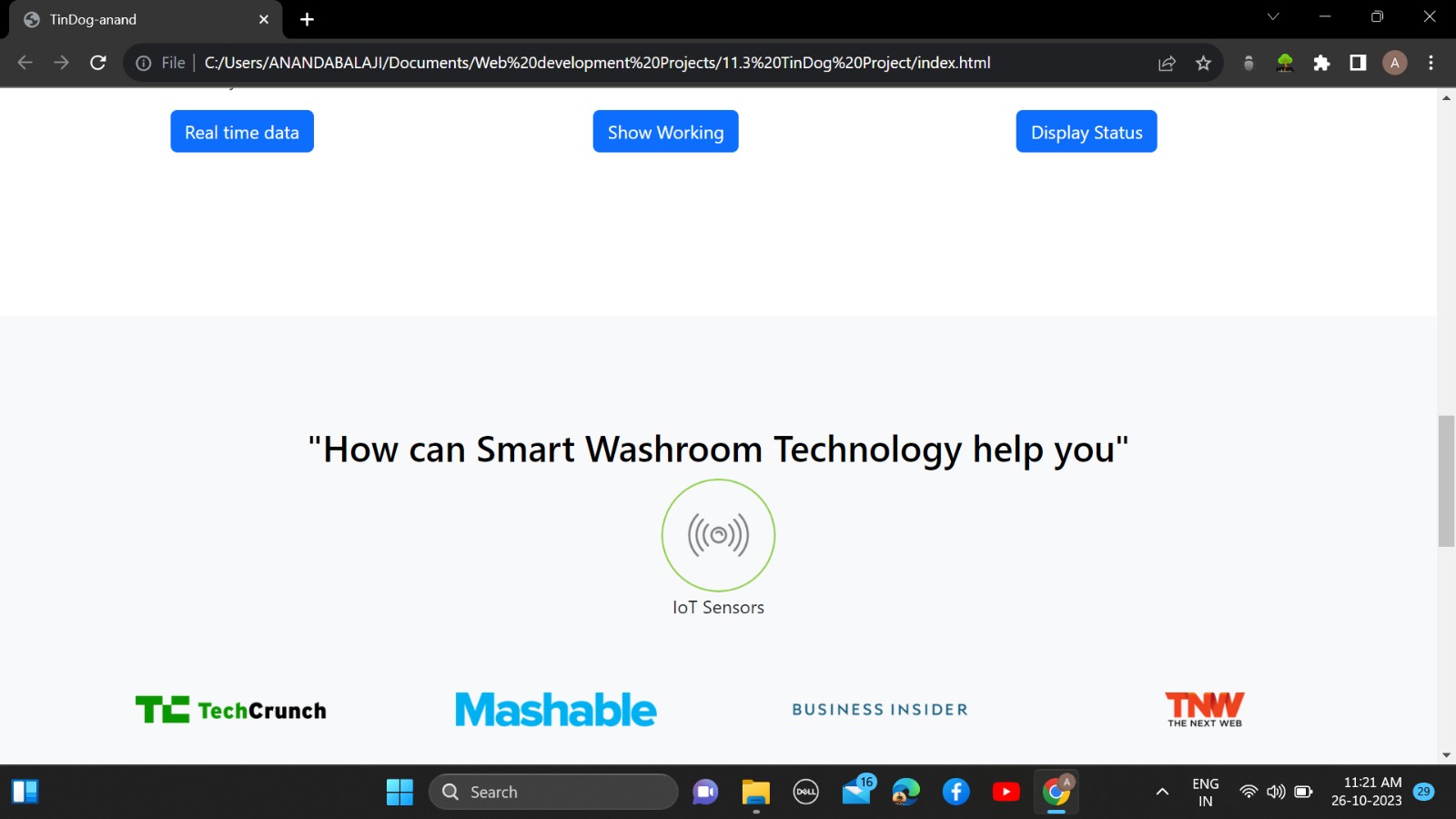


**Features:**

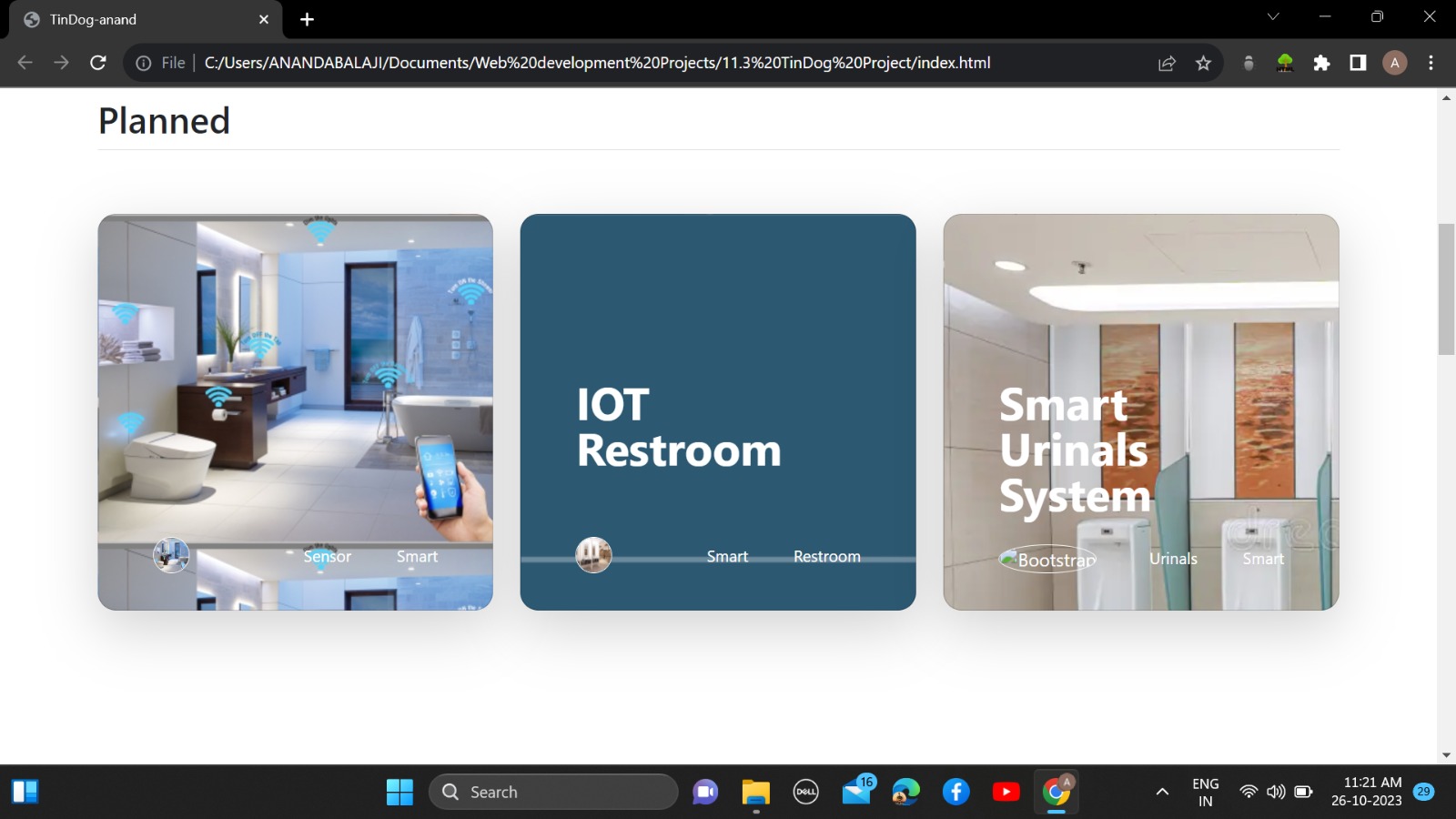








**Advancements:**



**11. Conclusion:**

The "Smart Public Fountain with IoT" project demonstrates the effective use of sensors for real-time data input, processing, and automation of fountain functions. The system uses occupancy, temperature, and humidity sensors to monitor conditions and act accordingly, ensuring a cleaner and more efficient public fountain experience. The digital display provides users with critical information about the fountain environment, enhancing their overall experience.

**12. References**

**- Paho MQTT Python Client - [https://pypi.org/project/paho-mqtt/](https://pypi.org/project/paho-mqtt/)**

**- Raspberry Pi GPIO -[https://pypi.org/project/RPi.GPIO/](https://pypi.org/project/RPi.GPIO/)**